

A Cross-Layer Approach for Improving TCP Performance in Mobile Environments

Deguang Le · Xiaoming Fu · Dieter Hogrefe

Published online: 13 November 2008

© The Author(s) 2008. This article is published with open access at Springerlink.com

Abstract Network-layer mobility protocols have been developed to keep continuous connectivity for mobile hosts while transparent to the higher layers. However, Due to its distinct characteristics of different from traditional TCP/IP environment, mobility poses substantial impacts on TCP performance in mobile environments. This paper proposes a new cross-layer approach, by introducing a mobility detection element in the network layer which interacts with the transport layer to optimize TCP operations. As changes are only made to the endpoints, this approach preserves the end-to-end semantics of TCP. Different from most exiting works, which utilize either transport or network layer alone without much cross-layer cooperation, our approach allows the use of mobility information in TCP. We analytically compare this approach against existing approaches and show that our approach outperforms prior approaches in terms of effective data resumption time. Through performance simulations, our approach demonstrates that it can effectively improve TCP performance in Mobile IPv6-based mobile environments.

Keywords TCP · Performance · Cross-layer · Mobility management · Mobile IPv6

1 Introduction

With the development of wireless access technologies, such as IEEE 802.11x, GPRS, 3G cellular networks, and Bluetooth, there is a trend converging these systems towards an IP-based infrastructure. To allow users with mobile devices to stay connected while moving to a new location, various mobility proposals have been developed [19]. One of the most mature solution is the IETF Mobile IPv6 (MIPv6) protocol [15], which handles mobility in the network layer. Various extensions for MIPv6 have been also developed, including the IETF Hierarchical Mobile IPv6 (HMIPv6) [27], Fast Handovers for Mobile IPv6 (FMIPv6) [18] and their cross-layer enhancements such as link layer triggers [21]. Although the MIPv6-

D. Le · X. Fu (✉) · D. Hogrefe
Institute of Computer Science, University of Göttingen, Lotzestr. 16-18, 37083 Göttingen, Germany
e-mail: fu@cs.uni-goettingen.de

based mobility approach makes the transport layer transparent from mobility, it may cause performance degradation in upper layers [2], especially for TCP [25], the key transport layer protocol.

TCP is a reliable transport protocol tuned to perform well in traditional fixed networks, where the route for each active TCP connection is relatively invariable and network congestion is the primary factor of affecting its performance. However, in mobile environments, it is common that the route of active connection changes more frequently due to communicating endpoints moving to a different location with different network characteristics. Without considering mobility, TCP will perform normal network congestion and trigger congestion control (e.g. exponential backoff of retransmission timeout or reduction of its congestion window) as done in fixed network cases, which may not be desired in mobile environments. This may result in significant end-to-end transport delay deviation and packet loss.

Existing works on TCP over wireless networks [5,8] focused on issues associated with wireless link features such as Bit Error Rate (BER) and loss, and usually do not consider the issue of mobility. Two broad approaches have been widely investigated: (1) link layer-based approach, e.g. SNOOP TCP [4]; (2) split connection approach, e.g. I-TCP [3], M-TCP [6]. These works assume the TCP over wireless problem as local problem and as a result attempt to address it locally.

Since support for mobility in IP networks has emerged as a critical issue, TCP performance issues over mobile environments using MIPv6 have been recently studied. Hsieh et al. [12] show that FMIPv6 and FMIPv6 in the network layer alone can bring better TCP performance than standard MIPv6. In the transport layer based approach, Fast Retransmission (FR) [7] and Freeze TCP [9] attempt to improve TCP performance by distinguish packet losses due to the host mobility from those due to the network congestion. FR [7] handles this by fast retransmission of the earliest unacknowledged data segments, immediately after completing the handover, while Freeze TCP [9] freezes the connection state as soon as the Mobile Node (MN) detects an impending handover. However, the trigger of reacting mechanisms to mobility in most of these approaches is rather dependent on the feedback from the transport layer itself, or the link signal strengths. None of these approaches could effectively and timely enough to detect network situation change (e.g. route change) in existence of any underlying mobility mechanism.

Demo-Vegas [11] is a TCP extension for performance improvements over Mobile IPv4 (MIPv4) networks, finely tuned for TCP Vegas without considering other possible TCP versions. Matsushita et al. [20] proposed modifications in both the mobile node and the access gateway, basically by applying the split connection approach to HMIPv6. Both approaches improve TCP performance to some extent, but do not sufficiently offer a general solution to improve TCP performance over mobile environments.

In this paper, based on a qualitative investigation on TCP over MIPv6 networks, we propose TCP-CM, a cross-layer approach utilizing the cooperation between network layer and transport layer for performing TCP reactively in mobile environments. TCP-CM can be easily extended for different versions of TCP (e.g. Vegas, Reno and Tahoe) and MIPv6 extensions (e.g. HMIPv6 and FMIPv6). Analysis shows that the effective data resumption time of TCP-CM is better than that of existing approaches. Simulation results show that our proposed approach can yield better performance than the standard TCP in MIPv6.

The remainder of this paper is organized as follows. Section 2 analyzes the problems of TCP related to mobility using MIPv6-based approaches. Section 3 describes our proposed approach. Section 4 presents analytical model to demonstrate the benefits of our approach over existing approaches, followed by simulation results in Sect. 5. Finally, conclusions and future work are provided in Sect. 6.

2 Problems Statement

The TCP protocol [25] is a general-purpose, reliable stream protocol. It works in different network environments, independent of medium, transmission rate, delay, and error rate. Standard TCP (TCP Reno) integrates support for acknowledgement, retransmission, congestion control, timing, maximum segment transmission mechanisms, and performs well in classic fixed networks. It primarily estimates retransmission timeout (*RTO*) based on measured round-trip time (*RTT*) and maintains two key parameters in the TCP sender: the congestion window (*CWND*) and the slow start threshold (*SSTHRESH*) for congestion control. In the case of MIPv6, when the MN moves and switches between the access networks, MIPv6 will perform the following procedures: movement detection, configuration of Care-of-Address (CoA) and binding registration. During these procedures, the communicating endpoints are not able to continue the communication between each other for some time (aka handover latency), which may however introduce the transport delay and packet loss, resulting in TCP congestion control mechanism in the following subtle ways. More details about MIPv6 and TCP can be found in [1, 17, 15, 24–26, 29].

Figure 1 visualizes the segment transmission from the Correspondent Node (CN) to the MN during the handover, where we assume that segment 1 is the last successful sent segment through the previous access network. From this figure, we can see that when the MN moves out of its previous network at time t_1 (i.e. the initiation of handover), because the TCP layer in the CN is unaware of the occurrence, it may continue to send segments to the MN’s previous network even if the MN has moved away (see Fig. 1 from segment 2 to segment k). Thus, any segments sent out by the CN from time t_0 (i.e. half RTT before the handover initiation) to time t_2 (i.e. the termination of handover) will not reach the MN. This causes that if the

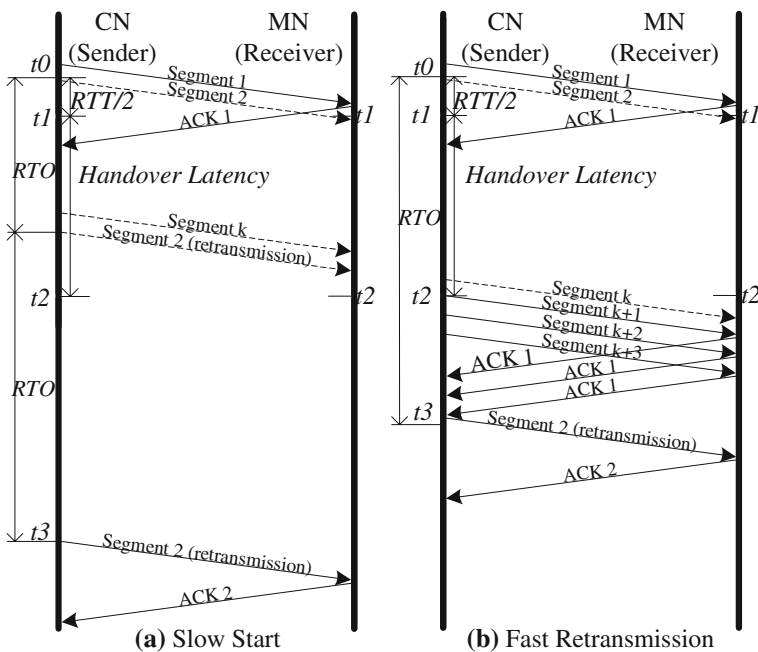


Fig. 1 Visualization of segment transmission from the CN to the MN during a MIPv6 handover

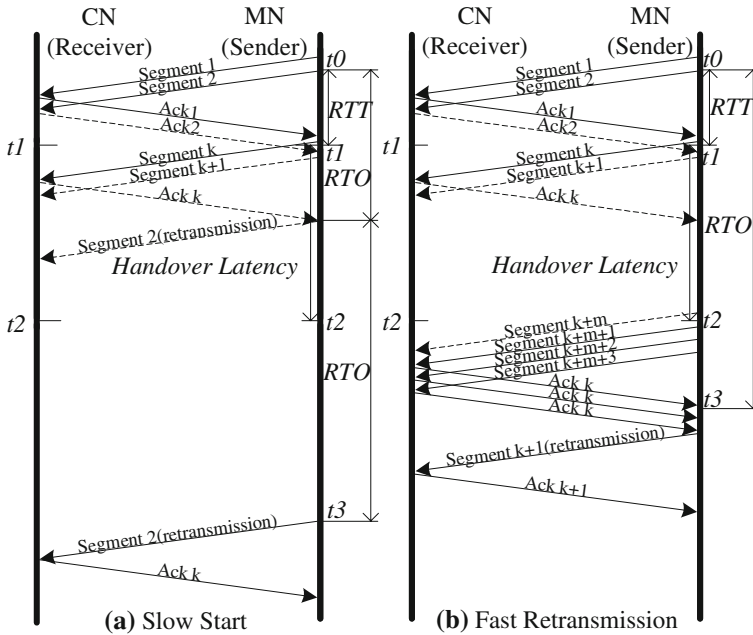


Fig. 2 Visualization of segment transmission from the MN to the CN during a MIPv6 handover

timeout occurs before the handover termination, the slow start congestion control will be invoked (see Fig. 1a), which will drop CWND to the size of one segment, and retransmits the earliest unacknowledged data segment (see Fig. 1a retransmitted segment 2); if the timeout does not take place during the handover latency (i.e. from time t_1 to t_2), the MN may receive the new data segment from the CN after the termination of handover if allowed by the current window size of the CN (see Fig. 1b from segment $k + 1$ to segment $k + 3$), which will incur the CN to retransmit the lost packets and begin the fast retransmission congestion control because of the consecutive duplicate acknowledgements from the MN (see Fig. 1b retransmitted segment 2).

The case for segments sending from the MN to the CN (see Fig. 2) is similar: when the MN moves out of its previous network, all the segments sent but not yet acknowledged in the sending window will not be acknowledged any more because of the unreachability of the acknowledgement messages, which are responded from the CN to the MN's previous network (see Fig. 2 from segment 2 to segment k). If the lost ACKs do result in a retransmission timeout, the MN will return to the slow start stage (see Fig. 2a), which reduces its CWND to one segment and retransmit the earliest unacknowledged data segment (see Fig. 2a retransmitted segment 2); if the timeout does not take place during the handover latency, the CN may receive the new segment from the MN after the handover termination if allowed by the current window size of the MN (see Fig. 2b segment $k + m + 1$). Then the packets sent before t_1 (see Fig. 2b from segment 2 to segment k) may be confirmed after the new acknowledgment is delivered due to the cumulative feature of ACKs. However, segments sent during the time from time t_1 to t_2 (see Fig. 2b from segment $k + 1$ to segment $k + m$) may be discarded by the interface of the MN during the movement detection and CoA configuration procedures or by the Home Agent (HA) during the stage of binding update procedure, which may likewise incur duplicated ACKs retransmission or timeout retransmission (see Fig. 2b).

In either scenario, the packet loss induces the timeout retransmission, finally causing a very small *CWND* and an equally small slow start threshold (*SSTHRESH*). Moreover, consecutive retransmission timeouts incur an artificial inflation of *RTO* due to Karn's exponential timer backoff algorithm [17]. As a result, the sender may waste long time to wait for a timeout before it can send new data continuingly (see Figs. 1a and 2a from t_2 to t_3). In case of duplicate ACKs retransmission (see Figs. 1b and 2b), even if *SSTHRESH* is dropped to half of their previous values, it may also be stale for the new network. For example, when the new network has a lower capability (i.e. Bandwidth Delay Product, *BDP*) than the previous network, the new setting may still be too high for the new network. The sender may inject more segments into the new network than necessary, which could cause the transmission queue at the bottleneck link's router to overflow. Conversely, if the new network is capable of carrying much more segments or is more lightly loaded than the previous network, the value of *SSTHRESH* will become irrelevant, and the increase of *CWND* is limited to one segment per *RTT*. It therefore will take multiple *RTT*s before reaching a reasonable throughput due to the slowness of additive increase. In addition, in MIPv6, when the MN moves from one network to another network, the route between communicating endpoints will change, causing the end-to-end transport delay to vary and the deviation of *RTT* increases quickly due to different network characteristics. Therefore, the traditional *RTO* update algorithm [24] does not adapt to the old *RTT* and its deviation to estimate the *RTO* in the new network, which may result in TCP to mistakenly make the congestion control decision based on invalid *RTT/RTO*.

3 A Cross-Layer Approach for Improving TCP Performance Over Mobile IPv6

The above analysis demonstrate that, in highly dynamic network environments, it is difficult to maintain high transport performance as well as extended mobility functionality via a single-layer solution. This motivates the design of a cross-layer approach, so-called TCP-MC, enabling TCP be more responsive to mobility behaviors in MIPv6. This approach introduces only modest changes to TCP and MIPv6 to enhance cooperation between them through inter-layer signalling interactions, allowing TCP to tune its behaviours reactively according to the procedures of the underlying mobility management protocol. The following subsections detail the key design rationales. We will see in Sect. 4 that the proposed approach is extensible for different MIPv6 extensions.

3.1 Enhancement of Inter-Layer Interaction Between TCP and MIPv6

To enhance the cooperation between TCP and MIPv6, MIPv6 must monitor the mobility-related procedures and is aware that subsequent procedures should be optimized by TCP. In addition, TCP also requires a way to learn of these mobility-related events of MIPv6.

We define two types of mobility events in MIPv6: Initiation_of_Handover (IoH) and Termination_of_Handover (ToH). The MN can easily identify its handover initiation through the Neighbour Unreachability Detection mechanism [22], through which the MN can actively probe the current access router using unicast Neighbor Solicitation messages to verify if the forwarding route is still working. If the MN receives a solicited Neighbor Advertisement with the Solicited flag set, it confirms that the current access router is reachable. Otherwise, it indicates the MN is moving out of its current network, causing the event of IoH. For the event of ToH, we can determine the event through the binding registration mechanism, through which the MN will receive the successful binding acknowledge when handover terminates. Following the mobility-related events, MIPv6 may signal TCP. The signalling may be delivered

through the Interface Control Information (ICI) [32], which allows the IP module to notify mobility events and triggers corresponding TCP operations.

At the CN, an additional communication step is necessary to inform TCP of the events occurring at the other endpoint (i.e. the MN) of the connection. For the notification of IoH, it is necessary for the MN to predict the reasonable time (i.e. a warning period) before which an actual Neighbour Unreachability occurs. Therefore, in our approach, after the notification of the completion of handover as described above, the MN forwards the signal of other events except the IoH over the new network to the CN. The signalling travels from the MN to the CN through the normal IP route and can choose either of the following two ways.

One way is to extend TCP, similar to the explicit notification mechanism by setting an Explicit Mobility Notification (EMN) flag [10] and introducing an Explicit Handover Notification bit [11, 13] in a specially marked TCP acknowledgement segment, or using a special option in TCP [30]. This mode simplifies some of the inter-layer issues at the CN. However, it has several drawbacks: the mobility notification may have to be redone for other transport protocols, it becomes difficult to share messages between different transport protocols, and the connection-oriented state maintained by TCP may not easily extend to save events for long periods. In addition, the inter-layer interaction of the MN is still inevitable.

Another way is to extend the IP layer to make use of the MIPv6 binding registration mechanism to transfer mobility-related events of the MN. We assume both the MN and the CN support the MIPv6 route optimization mode. hence this network-layer extension approach is preferred. By extending the Binding Update message with a ToH bit to indicate the CN's IP layer upon the detection of the mobility-related event, the event can then be transferred to TCP through an inter-layer interaction mechanism.

As soon as TCP is aware of the mobility-related events of the MN, it can react to mobility behaviors of the MN by tuning the state parameters. The adjustments of state parameters are derived from the mechanisms described in the following subsection.

3.2 TCP Activities to Mobility in MIPv6

When TCP is aware of the mobility-related behaviours of MIPv6, it invokes the following activities in response to mobility in MIPv6:

1. Congestion Avoidance: Prevent the invocation of congestion control while the MN moves. Since it is common that the communicating endpoints change their route while communicating with each other, it is not suitable that each time the route variation occurs, the window size starts from the slow start stage. Therefore, we need a way to optimize *CWND/SSTHRESH* values by quickly probing the route's available capacity. The optimal congestion window size can be determined based on the bandwidth estimation. In addition, updating the congestion window is based solely on ACKs for data sent through the new network.
2. Fast Recovery: Ignore the outstanding segments in the send window. These segments or their ACKs, which are sent before the handover initiation, are in flight. Although the retransmission timeout of these segments may not take place, they in fact can not reach the receiver any more. Therefore, these segments should be retransmitted immediately.
3. Timing Re-initialization: Reset the bogus timing parameters. It is obvious that the *RTT* fluctuates significantly in mobile environments. Traditional TCP computes the smooth *RTT* by using previous values and *RTO* depends on the smooth *RTT* [24]. In order to avoid false timeout and artificial inflation of pause, the timing parameters *RTT/RTO* should be reinitiated to the value of *RTT* acquired from the new network.

In [7], it simply retransmits the earliest unacknowledged segment immediately followed by the congestion control (i.e. the timeout) backoff. However, it can not prevent from invoking the congestion control procedure. In our approach, the unnecessary congestion control can be avoided through the mechanism (1) and (3). Besides, when the MN moves and the handover takes place, the data of TCP connection may pass through different routes over heterogeneous networks (i.e. mobile packet routing). Due to the different network characteristics, the parameters of TCP connection (e.g. *CWND*, *SSTHRESH*, *RTT* and *RTO*) for the old network may not be suitable to the new network, so they need be tuned timely along with the changes of attachment networks. In [9], the Freeze TCP chokes all retransmission timers and enters persist mode without shrinking its *CWND/SSTHRESH* size through the Zero Window Advertisement (ZWA) when the handover takes place. When the MN moves into the new network, the TCP connection resumes through the Triplicate Re-connection ACKs (TR-ACKs) mechanism as suggested in [7]. This simple tuning of TCP parameters can not match the new network characteristics, especially with frequent handovers. Our above congestion avoidance and timing re-initialization together are employed to handle differences in network characteristics upon mobility events by optimizing *CWND/SSTHRESH* values and re-initializing the timing variables *RTT/RTO*. Ho et al. [11] simply does the mechanism (1) by employs Vegas to adjust the congestion window for congestion avoidance in the new network after handover. However, it does not consider stale data segments in retransmission queue and bogus timing. Thus, it is incapable of handling artificial inflation of *RTO*. Through the combination of above mechanisms, as also shown in the simulations in the following section, the adverse impacts on TCP introduced by the mobility features of MIPv6 can be overcome effectively in a timely fashion, which in turn prevent the degradation of transport performance.

After describing the changes for TCP, the following subsections describe the mechanisms required for different types of MIPv6 entities.

3.2.1 Activities for Mobile Node

Consider the scenario where the MN is sending data to the CN. When the MIPv6 of the MN determines that the current access router is no longer reachable through the neighbor unreachability detection mechanism, it notifies TCP of the event of IoH. TCP then stops sending segments to the IP layer and the counter of *RTO* is suspended immediately. When the MN receives a packet of valid Binding Acknowledgement, which indicates the handover completion, it notifies TCP of the event of ToH. Once TCP receives the signal of ToH, it resumes the *RTO* value and TCP may resume the transmission if allowed by the sending window size. If the usable sending window size is zero, the window size is increased by one segment, and then TCP goes on transmitting the segment to the CN. Therefore, the unacknowledged segments to the CN sent through the previous network may be confirmed after the new acknowledgment through the new network is delivered due to the cumulative feature of ACKs.

3.2.2 Activities for Correspondent Node

Consider the second scenario where the CN is sending data to the MN: as soon as TCP at the CN receives the signal of ToH, it clears the *SRIT* (i.e. the smoothed *RTT*) and *RITVAR* (i.e. the *RTT* variation) and resets *RTO* according to the following formulas [24].

$$SRIT = R \quad (1)$$

$$RITVAR = R/2 \quad (2)$$

Table 1 Effective data resumption time

Protocol	Effective data resumption time
FMIPv6	$4T_{CN-oAR} + 2T_{oAR-nAR}$
HMIPv6	Intra-domain: $4T_{CN-MAP} + 2T_{MAP-oAR} + 2T_{MAP-nAR}$ Inter-domain: $2T_{CN-oAR} + 4T_{CN-MAP} + 12T_{MAP-nAR} + 4T_{HA-MAP}$
FR	$T_{CN-oAR} + 3T_{CN-nAR} + 2T_{HA-nAR}$
Freeze TCP	$T_{CN-oAR} + 3T_{CN-nAR} + 2T_{HA-nAR}$
Mobile TCP	$T_{CN-oAR} + 3T_{CN-nAR} + 2T_{HA-nAR}$
TCP-CM	$T_{CN-oAR} + 2T_{CN-nAR} + 2T_{HA-nAR}$
TCP-CM o.FMIPv6	$2T_{CN-oAR} + 2T_{oAR-nAR}$
TCP-CM o.HMIPv6	Intra-domain: $4T_{CN-MAP} + 2T_{MAP-oAR} + 2T_{MAP-nAR}$ Inter-domain: $2T_{CN-oAR} + 2T_{MAP-nAR} + 2T_{MAP-nAR} + 2T_{HA-MAP}$

$$RTO = SRTT + \max(G, K * RITVAR) \tag{3}$$

where R is the value of RTT observed over the new network, G is the granularity of the clock, and K is a coefficient.

Since the outstanding segments in the retransmission queue travel through the previous network, and will never reach the MN, these segments are retransmitted, but not invoke the congestion control mechanism. Besides, the CN resets $CWND$ and $SSTHRESH$ according to the BDP of the new network, which represents a sender’s TCP congestion window [29]. That is, the $SSTHRESH$ is set equal to the available pipe size when connections are switched to the new network (i.e. $BWE * RIT_{new}$), and $CWND$ is set to be equal to $SSTHRESH$. Then, the congestion avoidance takes over, where BWE means the estimated bandwidth and the value RIT_{new} is measured over the new network.

4 Analysis and Comparison

Our proposed approach has several differences from previous approaches as discussed above. In the following subsections, we will identify its advantages in terms of effective data resumption time introduced by mobility based on a mathematical model.

4.1 Considered Model

We firstly introduce a MIPv6 topology model used for analytical study. Figure 3 depicts the considered model. We assume that a Mobile Node (MN) is initially located at the old Access Router (oAR) and then moves from the oAR to a new AR (nAR). The mobile node receives the data packets sent from the Correspondent Node (CN). The MAP refers to the mobility anchor point employed by HMIPv6 for hierarchical mobility management [27]. The analysis will study the network-layer approaches (i.e. FMIPv6 [18] and HMIPv6 [27]), transport-layer approaches of FR [7], Mobile TCP [28] and Freeze TCP [9], and our proposed TCP-MC. The reason is that all of them can handle the packet loss during the effective data resumption time introduced by the handover as discussed in Sect. 2 (see Table 1).

4.2 Assumptions and Parameters

In order to compute the effective data resumption time we have to consider the delay introduced by packet transmission between nodes. The effective data resumption time will be

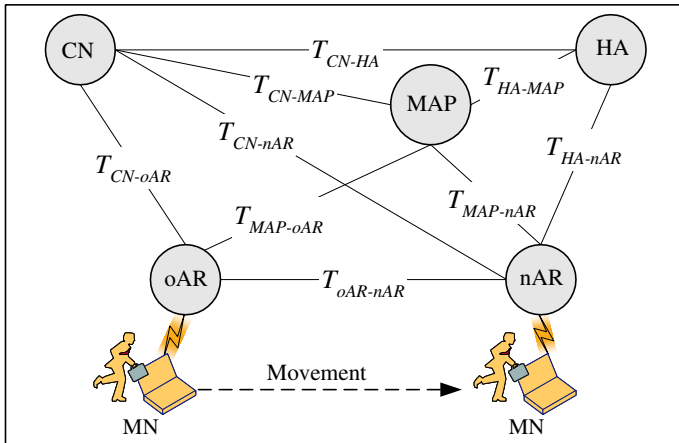


Fig. 3 Considered scenario for numerical analysis

analyzed considering the mobile node initiated handover case. We assume that the processing delay are negligible compared to access to the channel and transmission delays. With respect to the parameters, we have the following assumptions.

- T_{i-j} only denotes the transmission delay of path between any two entities i and j . For example, T_{CN-oAR} is referred to as the time required by forwarding packets on the path from the CN to the oAR, or reversely. We suppose the same value for both uplink and downlink directions.
- The processing latency of local trigger in a mobile node’s protocol stack is ignored. Thus, the period used to receive a moment hint with link-layer support is zero.
- The latency of wireless links between a mobile node and access router is ignored since it is much smaller than that of transmission delay on the global Internet.

4.3 Numerical Analysis

In MIPv6, if the network prefix has changed, the MN need configure a new temporary CoA. Then, the MN updates its location by advertising its new CoA with its HA and eventually its CN(s). In general, the handover procedures can be regarded to comprise three blocks: the movement detection, the CoA configuration and the home (or correspondent) registration. During the handover procedures, there is a period when the MN is unable to receive TCP data segments both due to link switching and TCP/IP protocol operations, called TCP payload discontinuity for TCP connection. Therefore, we define the effective data resumption time of TCP connection as follows:

Definition of Effective Data Resumption Time The effective data resumption time of TCP connection refers to the time period between the receipt of the last TCP data segment on a previous access network and the resumption receipt of a new TCP data segment on the new access network when a communicating endpoint moves and changes its access networks.

Based on this definition, the overall effective data resumption time of TCP connection, which is represented by T , may thus be computed as the sum of the time steps as follows:

$$T = T_1 + T_2 + T_3 \tag{4}$$

Here T_1 refers to the transmission time that the TCP data segment is sent by the CN to the previous network of the MN before the handover initiation; T_2 refers to the time period that

elapses during the handover procedures. It includes the movement detection time, CoA configuration time and binding update time; T_3 refers to the time of RTO for TCP data segment lost during the handover after the handover termination, or the time for duplicated TCP data segment transmission after the handover termination.

4.3.1 TCP-CM

Considering our proposed model described in Fig. 3, the effective data resumption time of T_{TCP-CM} will be the sum of the following contributions: (1) the time that any segment is sent by the CN to the old network of the MN before the handover initiation (i.e. T_{CN-oAR}); (2) the time the MN receive Unsolicited Router Advertisement (URA) messages from the access routers, which indicates that an MN entering a new link (i.e. the movement detection and handover initiation) [22]; (3) the time that the MN sends Router Associated (RA) messages for stateless address auto-configuration [31] over wireless link; (4) the time required to send the BU and receive the BA through the wireless medium for home registration; (5) the time required by the BU to reach the HA who forwards the packet to the MN (T_{HA-nAR}); (6) the time required by the BA to reach the current nAR (T_{HA-nAR}); (7) the time required to send the BU and receive the BA through the wireless medium for correspondent registration; (8) the time required by the BU to reach the CN who continuously sends the packet to the MN (T_{CN-nAR}), and (9) the time required by the BA to reach the current nAR (T_{CN-nAR}).

Based on our assumption, we ignore the part of local wireless link delay. Thus, the T_{TCP-CM} due to performing a handover from the oAR to the nAR in MIPv6 can be computed through the following equation:

$$T_{TCP-CM} = T_{CN-oAR} + T_{HA-nAR} + T_{HA-nAR} + T_{CN-nAR} + T_{CN-nAR} \quad (5)$$

4.3.2 FR

Considering the case of FR [7], as soon as the connection is re-established through the nAR, the MN sends triplicate ACKs without waiting for a retransmission timeout for the last data segment which it received prior to the discontinuity through the oAR [7]. When the CN receives three copies of ACK, it will continuously transit TCP segments. Therefore, the effective data resumption time of T_{FR} for FR is composed of the following parts: (1) the time required to successfully send the last TCP segment through oAR to the MN; (2) the time required by the MN performing the handover procedures, and (3) the time required by the TR-ACKs from the nAR to the CN.

Since the first two steps is calculated in Eq. 5, the T_{FR} performing a handover from the oAR to the nAR in FR can be calculated through the following expression:

$$T_{FR} = T_{CN-oAR} + T_{HA-nAR} + T_{HA-nAR} + T_{CN-nAR} + T_{CN-nAR} + T_{CN-nAR} \quad (6)$$

4.3.3 Freeze TCP

Freeze TCP has the same effective data resumption time as that of FR because it employs the same TR-ACKs mechanism suggested in FR [9]. Therefore, we get the T_{TCP-CM} as follows:

$$T_{FreezeTCP} = T_{CN-oAR} + T_{HA-nAR} + T_{HA-nAR} + T_{CN-nAR} + T_{CN-nAR} + T_{CN-nAR} \tag{7}$$

4.3.4 Mobile TCP

Considering the case of Mobile TCP [28], when an MN changes its access routers (e.g. movement from the oAR to nAR), the `icot1` call corresponding to the downing of an interface or changing of its address is trapped by the kernel device control code. All the sockets that use the affected interface are notified, and a switch detection routine is invoked. This causes the MN to notify the CN via an option field in the next Mobile TCP packet [28]. Therefore, an additional timeliness is required to be considered beside the time required to successfully send the last TCP segment through the oAR to the MN and the time required by the MN performing the handover procedures, namely the time required for explicit handover notification through a special TCP option.

Thus, the $T_{MobileTCP}$ due to performing a handover from the oAR to the nAR in MIPv6 can be expressed through the following equation:

$$T_{MobileTCP} = T_{CN-oAR} + T_{HA-nAR} + T_{HA-nAR} + T_{CN-nAR} + T_{CN-nAR} + T_{CN-nAR} \tag{8}$$

4.3.5 FMIPv6 and HMIPv6

Based on above analysis, we find that the main contributions of effective data resumption time come from the MIPv6 handover latency. Next, we study the effective data resumption time under the FMIPv6 and HMIPv6, which tries to optimize the TCP performance through decreasing the handover latency of MIPv6 (see Sect. 3).

FMIPv6 allows a MN to configure a new CoA before it moves and connects to a new network. It also allows the MN to use the new CoA immediately upon connecting to the new network. Besides, the FMIPv6 provides a bi-directional tunnel between the old and new networks for packets forwarding while the BU procedures are being performed. Thus, compared to standard MIPv6 operations, the FMIPv6 not only can remove IPv6 configuration delay introduced by movement detection and CoA configuration, but also eliminate the delay involved during the MN’s BU procedures. However, due to the features of timing and re-transmission of TCP, it is not necessary that the real effective data resumption time can be reduced in practice.

The RTO can be calculated through the algorithm of TCP’s retransmission timer [24] when the MN is attached to the oAR as follows:

$$RTO \approx 2T_{CN-oAR} \tag{9}$$

During the handover, the data packets will be forwarded between the oAR and the nAR through IP tunneling in FMIPv6 [18], hence the data transmission time between the CN and MN will be $T_{CN-oAR} + T_{oAN-nAR}$. Therefore, the inequality $2 * (T_{CN-oAR} + T_{oAN-nAR}) > RTO$ will be satisfied according to Eq. 9, which causes re-transmission timeout events.

Because of the features of timing and re-transmission of TCP, when a timeout occurs, the TCP sender will retransmit data segments and reset the retransmission timer to backoff interval according to Karn’s exponential backoff algorithm [17]. Therefore, the TCP sender (e.g. the CN) has to wait for yet another transmission of duplicate data segments before it can continuously transmit new data segments.

Thus, the T_{FMIPv6} due to performing a handover from the oAR to the nAR in FMIPv6 can be expressed through the following equation:

$$T_{FMIPv6} = T_{CN-oAR} + T_{CN-oAR} + T_{CN-oAR} + T_{oAR-nAR} + T_{oAR-nAR} + T_{CN-oAR} \tag{10}$$

HMIPv6 [27] is another extension to MIPv6 for optimizing the handover latency by introducing a new entity of Mobility Anchor Point (MAP) severing as a local home agent. When the MN moves within the MAP domain (called intra-domain mobility), mobility is handled by the MAP, and the MN only performs the BU procedures (i.e. local BU) to local MAP, which then eliminate the global BU procedures with its HA and CNs. However, when the MN moves between the MAP domains (called inter-domain mobility), the MN will have to perform both local BU procedures with the local MAP and global BU procedures with its HA and CNs. Therefore, the real effective data resumption time in HMIPv6 depends on the movement modes (i.e. intra-domain mobility and inter-domain mobility).

For intra-domain mobility, the handover latency ($T_{HO-Intra-Domain}$) will be the time required for the MN to send the local BU and receive the BA from the MAP:

$$T_{HO-Intra-Domain} = 2T_{MAP-nAR} \tag{11}$$

Similarly, we can calculate the RTO in MIPv6 before the handover initiation as follows:

$$RTO \approx 2T_{MAP-oAR} + 2T_{CN-MPA} \tag{12}$$

Since the MAP is located at the same local domain as the oAR and nAR, we assume that $(T_{MAP-oAR}) \approx (T_{MAP-nAR})$. Therefore, we can get $T_{HO-Intra-Domain} < RTO$ based on Eqs. 11 and 12. Because the TCP segments send out by the CN during the handover are lost, the retransmission timeout will occurs and the lost TCP segment will be retransmitted. Thus, the effective data resumption time in the intra-domain mobility is calculated as follows:

$$T_{HMIPv6-Intra-Domain} = T_{CN-MAP} + T_{MAP-oAR} + T_{MAP-oAR} + T_{CN-MAP} + T_{CN-MAP} + T_{MAP-nAR} + T_{MAP-nAR} + T_{CN-MAP} \tag{13}$$

For inter-domain mobility, the handover latency ($T_{HO-Inter-Domain}$) will be the time required for the MN to send the local BU and receive the BA from the MAP, and plus the time required for global home registration and correspondent registration:

$$T_{HO-Inter-Domain} = T_{MAP-nAR} + T_{MAP-nAR} + T_{MAP-nAR} + T_{HA-MAP} + T_{HA-MAP} + T_{MAP-nAR} + T_{MAP-nAR} + T_{CN-MAP} + T_{CN-MAP} + T_{CN-MAP} + T_{MAP-nAR} \tag{14}$$

From the Eqs. 12 and 14, we get $T_{HO-Inter-Domain} > RTO$, which causes that the handover is not yet complete when the timeout occurs. Thus, during the handover, the retransmitted TCP segments are also lost, resulting consecutive timeouts. Based on multiple backoffs of the retransmission timer, the effective data resumption time will double the handover latency. Then, the effective data resumption time in the inter-domain mobility of HMIPv6 can be expressed as follows:

$$\begin{aligned}
 T_{HMIPv6-Inter-Domain} = & 2 * (T_{CN-oAR} + T_{MAP-nAR} \\
 & + T_{MAP-nAR} + T_{MAP-nAR} \\
 & + T_{HA-MAP} + T_{HA-MAP} \\
 & + T_{MAP-nAR} + T_{MAP-nAR} \\
 & + T_{CN-MAP} + T_{CN-MAP} \\
 & + T_{MAP-nAR})
 \end{aligned} \tag{15}$$

4.3.6 TCP-CM Over FMIPv6 and HMIPv6

TCP-CM is a flexible approach and as such it is also applicable for MIPv6 extensions such as FMIPv6 and HMIPv6. In the case of these extensions, TCP-CM also enjoys benefits of FMIPv6 and intra-domain HMIPv6 such as reduced handover time.

Following the above analysis, the effective data resumption time for TCP-CM over these MIPv6 extensions could be similarly derived as following equations:

$$T_{TCP-CM-FMIPv6} = T_{CN-oAR} + 2T_{oAR-nAR} + T_{CN-oAR} \tag{16}$$

$$T_{TCP-CM-HMIPv6-Intra-Domain} = 2T_{CN-MAP} + 3T_{MAP-nAR} \tag{17}$$

$$\begin{aligned}
 T_{TCP-CM-HMIPv6-Inter-Domain} = & 2T_{CN-oAR} + 2T_{MAP-nAR} \\
 & + 2T_{MAP-nAR} + 2T_{HA-MAP}
 \end{aligned} \tag{18}$$

4.4 Numerical Results

To summarize the above analysis, the effective data resumption time introduced by TCP-CM, FR, Freeze TCP, Mobile TCP, FMIPv6 and HMIPv6, as well as TCP-CM over FMIPv6 and HMIPv6, are represented in Table 1.

First, we compare the effective data resumption time introduced by network-layer approaches (i.e. FMIPv6 and HMIPv6) with that of our proposed scheme. Then, the effective data resumption time caused by transport-layer approaches (FR, Freeze TCP and Mobile TCP) will be compared with that of our proposed scheme.

In the analytical model illustrated in Fig. 3, we assume the paths between to entities, i.e. from the CN to the oAR, from the CN to the nAR, from the CN to the HA, from the CN to the MAP, from the oAR to the nAR, from the HA to the MAP, and from the HA to the nAR, are parts of the Internet since these entities are independent of each other from the geographical location point of view. Without the loss of generality, the transmission time of these pathes is considered the same as the delay of successively-transmitted Internet packets. We denote the total delay in the Internet as $T_{Internet}$ [16], so we can get the following expression:

$$\begin{aligned}
 T_{Internet} \approx & T_{CN-oAR} \approx T_{CN-nAR} \approx T_{CN-HA} \\
 \approx & T_{CN-MAP} \approx T_{oAR-nAR} \approx T_{HA-MAP} \approx T_{HA-nAR}
 \end{aligned} \tag{19}$$

Besides, since the MAP is located at the same local domain as the oAR and nAR, we assume the following expression reasonably:

$$T_{Internet} > T_{MAP-oAR} \approx T_{MAP-nAR} \tag{20}$$

Thus, based on Eqs. 16 and 17, the effective data resumption time of FMIPv6 will be mainly produced by six times of the delay that the data packet passes through the Internet, i.e. $T_{FMIPv6} = 6 * T_{Internet}$. For HMIPv6, the effective data resumption time will be mainly produced by only over four times when the data packet passes through the Internet in this case of intra-domain mobility, i.e. $T_{HMIPv6} > 4 * T_{Internet}$, but over ten times in this case of inter-domain mobility, i.e. $T_{HMIPv6} > 10 * T_{Internet}$. Comparatively, it takes only $5 * T_{Internet}$ for our TCP-CM approach, so it can be easily understood that TCP-CM has lower effective data resumption time compared with the network-layer approaches.

Considering the transport-layer approaches, it is very easy to get all of the FR, Freeze TCP and Mobile TCP will lead to at least $6 * T_{Internet}$ of effective data resumption time, which is also larger than that of our TCP-CM.

It is also shown that TCP-CM over MIPv6 extensions such as FMIPv6 and HMIPv6 performs better than the standard TCP-CM, FMIPv6 or HMIPv6 alone.

The above analytical results indicate that TCP-CM outperforms both the network-layer approaches and transport-layer approaches in terms of the effective data resumption time. In the next section, we will further evaluate our approach by means of simulation.

5 Simulation and Evaluation

In this section, we evaluate the effect of the proposed mobility control mechanisms through simulations performed using OPNET simulator [23]. The simulation models are built by extending the mobility control mechanisms from the standard MIPv6 and TCP model. The simulation results demonstrate how the enhancement of TCP performance can be achieved in our approach.

5.1 Model Modifications

Like the traditional TCP/IP model, in the OPNET model, the TCP and IP modules are not allowed to transmit control information except the data delivery. To enhance inter-signalling communication, we implemented a layered protocol interfacing between the TCP and IP modules through the Interface Control Information (ICI) mechanism. The ICI allows the IP module to notify mobility events and incurs corresponding mobility controls.

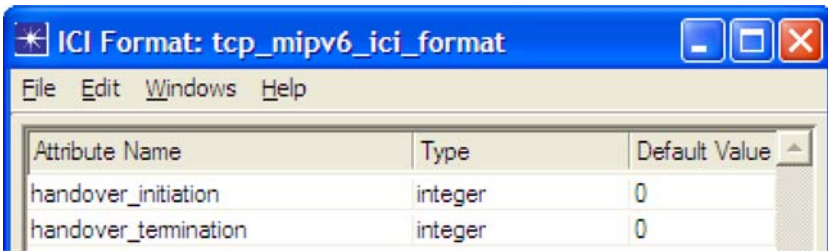
5.1.1 ICI Format Specification

An ICI format defines the structure of an ICI in terms of the attributes that it contains. An attribute's data type determines what sort of information it can store. In our ICI contents, we specify the following attribute names: `handover_initiation` and `handover_termination`, where, `handover_initiation` means the event that the MN determines that the current access router is no longer reachable through Neighbour Unreachability Detection mechanism; `handover_termination` means the MN receives a packet with a valid Binding Acknowledgement. The format specifications for ICI between IP and TCP are shown in Fig. 4.

5.1.2 ICI Mechanics and its Implementation

The sequences of operation for the TCP and IP modules are as follows:

We create a new ICI at the IP module using the Kernel Procedure (KP) `op_ici_create()`, and supply the ICI format specified in previous Subsection. Then we associate the created



Attribute Name	Type	Default Value
handover_initiation	integer	0
handover_termination	integer	0

Fig. 4 ICI format specification

ICI with the events of IoH and ToH respectively by calling the KP `op_ici_install()`. When the IP module is about to execute processes that will generate the events described above, and the ICI is to be associated with the corresponding events, it assigns correct values to the ICI's attributes by calling the KP `op_ici_attr_set()`. The following code for the indication and notification of IoH is implemented in the *away* state of the `mipv6_mn` process model:

```
node_objid = op_topo_parent (op_id_self ());
tcp_mod_objid = op_id_from_name (node_objid, OPC_OBJTYPE_PROC, "tcp");
if (aye_handover_ici_ptr == OPC_NIL)
{
    aye_handover_ici_ptr = op_ici_create ("aye_handover_ici_format");
} endif
op_ici_attr_set (aye_handover_ici_ptr, "aye_handover_initiation", 1);
op_ima_obj_attr_get (compound_attr_objid, "Route Optimization", &aye_opt_flag);
if (aye_opt_flag)
{
    op_ici_attr_set (aye_handover_ici_ptr, "aye_route_optimization", 1);
} endif
if (!L3_HO_HOME)
{
    op_ici_attr_set (aye_handover_ici_ptr, "aye_l3_mobility_away", 1);
} endif
op_ici_install (aye_handover_ici_ptr);
op_intrpt_force_remote (AYE_MOBILITY_CONTROL_INTRPT_CODE, tcp_mod_objid);
```

When the indicated events occur and result in an interrupt, the interrupted process in the TCP module obtains the ICI by calling the KP `op_intrpt_ici()`. TCP extracts information that it needs from the ICI by calling the KP `op_ici_attr_get()`, and responds correspondingly. Figure 5 shows the extended state diagram of the TCP process model. In this figure, we create a new state of `MOB_CTR` and define the condition of `MOBILITY_CONTROL`, which is associated with mobility-related events. As soon as the condition is true, it will transit to the `MOB_CTR` state, and perform corresponding Enter Executions as described below.

The following code is implemented in the extended `MOB_CTR` state of the TCP process model:

```
if (op_ici_attr_get (aye_ici_ptr, "handover_initiation", &tcp_
handover_initiation) == OPC_INITIATION_OF_HANDOVER)
{
```

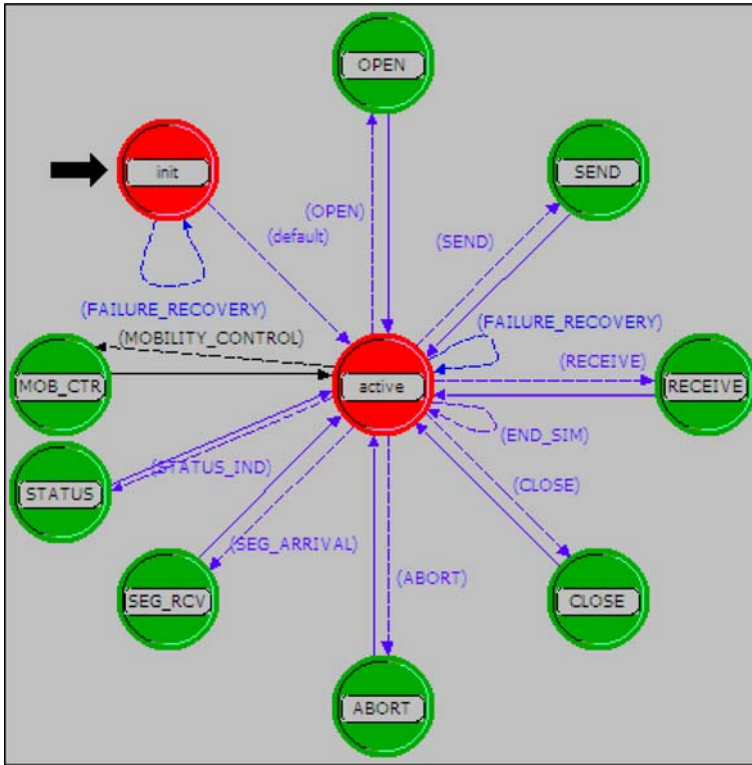


Fig. 5 Extended TCP process model

```

current_time = op_sim_time ();
avail_wnd = 0;
} endif
if (op_ici_attr_get (aye_ici_ptr, "handover_termination",
&tcp_handover_termination) == OPC_TERMINATION_OF_HANDOVER)
{
retrans_rtt = measured_rtt;
retrans_rtt_dev = measured_rtt/2;
current_rto = retrans_rtt + rtt_dev_coef*retrans_rtt_dev;
ssthresh = (bw*retrans_rtt)/snd_mss
cwnd = ssthresh
snd_wnd = int(snd_wnd/snd_mss)*snd_mss
if (snd_wnd<=snd_next-snd_una)
{
snd_wnd = snd_wnd+snd_mss
} endif
next_timeout_time = next_timeout_time - (op_sim_time ()-
current_time);
avail_wnd = snd_una + total_wnd - snd_next;
} endif
    
```


5.2 Experimental Setup

To evaluate the effect of our approach, the TCP performance test is carried out in our mobile IPv6 test environments as illustrated in Fig. 6. The simulation network topology model is composed by the IP cloud model, Access Network (AN) model, Access Routers (ARs), and communicating endpoints (i.e. the MNs and the CNs) etc. The IP cloud model represents the Internet through which the IP traffic can be modelled. The access network model represents the access network technology with different features and capacities. AR represents the wireless access base station of access networks. In addition, each AR consists of two interfaces, among which the wired interface is connected to the access network model through wired link and the wireless interface running IEEE802.11b with a coverage area of approximately 300m in radius provides wireless access for the MNs. The HA (i.e. home AR) represents the home agent with home agent function, while the AR2, AR3 and AR4 (i.e. foreign ARs) represent the foreign access router of AN2, AN3 and AN4 respectively. Each AR is positioned to be 250m apart with free space each other to ensure the overlapping distance. For simulation operation, we define a FTP file transfer application through the Application object and Profiles object. The concrete application configuration will be described in the following subsections in detail.

In this simulation scenario, the MNs (i.e. the MN_FTP_Server and MN_FTP_Client) and the CNs (i.e. the CN_FTP_Server and CN_FTP_Client) communicate to each other by running a ftp application as a source of TCP traffic. Initially, the MNs are placed at their corresponding home networks, and the MNs move in a counterclockwise trajectory roaming through all four access networks on the Internet, so that three general movement modes, namely movement from home network to foreign network (see Fig. 6 movement from access network1 to access network2), movement among foreign networks (see Fig. 6 movement from access network2 through access network3 to access network4), and movement from foreign network to home network (see Fig. 6 movement from access network4 to access

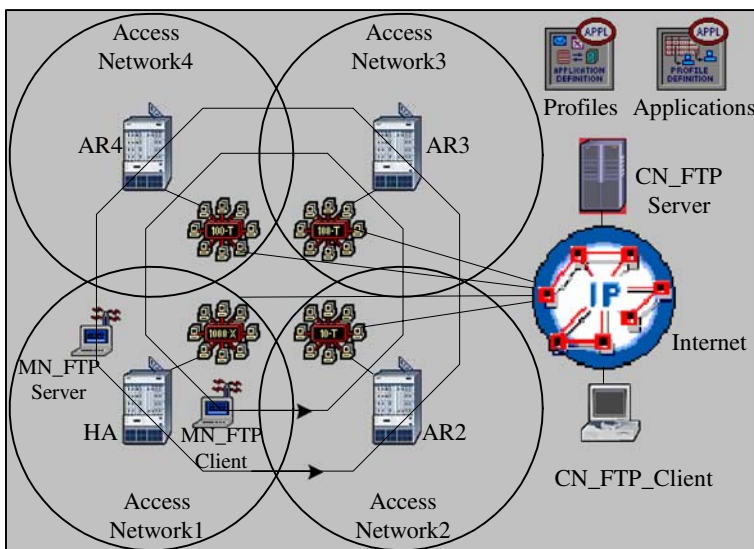


Fig. 6 Simulation network model

network1), can be observed in simulations. The MNs are served by the home agent (e.g. the HA) when they moves into the foreign networks.

5.3 Measurements and Results Evaluation

In this subsection, we measure the adaptive TCP behaviours for standard MIPv6, including the received sequence number, RTO, and CWND, to verify the correctness of our approach. Besides, we also investigate the performance of throughput to validate the effect of our approach.

5.3.1 TCP Parameter Measurements and Evaluation

First we measure the sequence number transmitted between the MN and CN by running a FTP application. In our simulation, we use the standard TCP Reno model and set the MN_FTP_Server as the FTP server and the CN_FTP_Client as the FTP client. The FTP server was sending a file of 16Mbytes to the CN_FTP_Client while it moves in the range of ANs. In addition, we set the buffer size of ARs larger than the send window of TCP in order to avoiding the packet loss due to the buffer overflow of link layer, which eliminates the impact of link layer factor.

Figure 7 depicts the sequence number of the received TCP segments versus time for one of the experiments, from which the comparison of sequence number progression in a connection using our MIPv6-aware TCP and a connection using unmodified TCP can be observed. It can be seen from the figure that our proposed approach recovers fast than the traditional TCP/MIPv6. This is because that we use the fast recovery mechanism, through which the TCP sender transmits data segments as fast as possible without waiting for an unnecessary timeout.

The retransmission timeout will be directly affected by the RTT, including SRTT and its deviation RTTVAR. In the standard MIPv6, the successive retransmission timeouts result in the artificial inflation of RTO. Figure 8 illustrates the growth and variety of RTO versus the simulation time when the endpoints are communicating with each other while the MN is roaming. From this figure, we can see that at about time 50s when the MN start switching

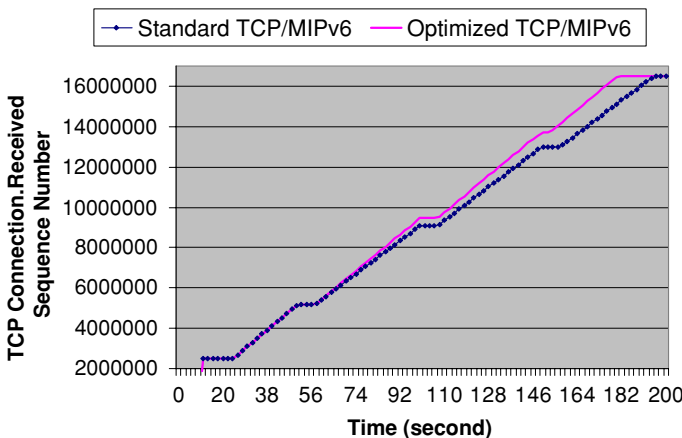


Fig. 7 Optimized TCP sequence number in MIPv6

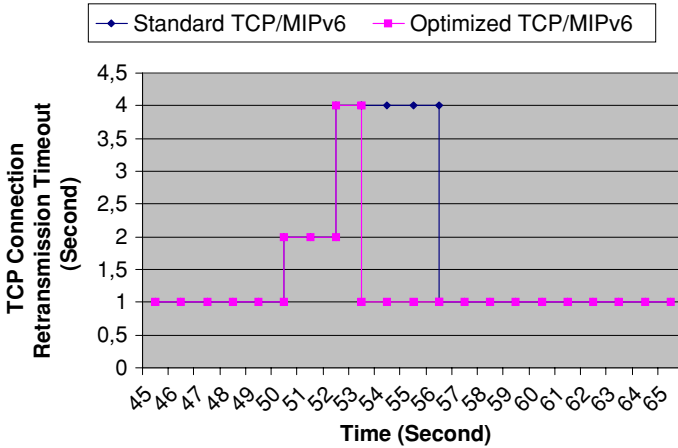


Fig. 8 Optimized TCP RTO in MIPv6

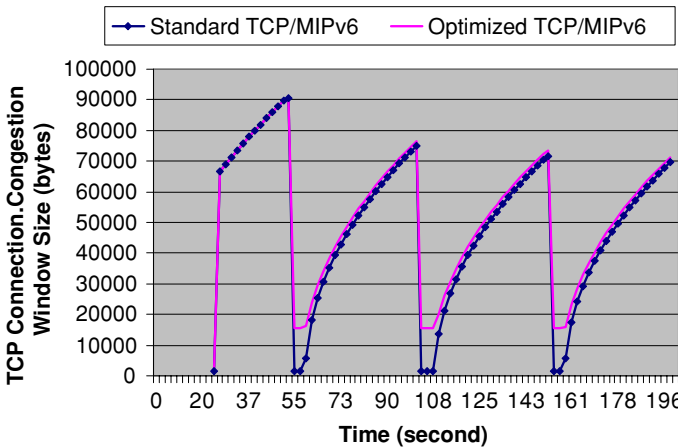


Fig. 9 Optimized TCP CWND in MIPv6

to a new network (i.e. the handover initiation), the RTO grows exponentially due to consecutive retransmission timeouts and exponential timer backoff algorithm [24]. When the MN completes the handover at about time 53 s (i.e. the handover termination), with the standard TCP/MIPv6, the TCP has to wait for a long time before it could continue sending the segment due to the artificial inflation of RTO. However, with our approach proposed in above section, the RTO is reduced significantly compared to the standard TCP/MIPv6 as soon as the handover termination. This is because, in our approach, the TCP can be aware of the handover termination timely, and it then could reinitiate the SRTT and RTTVAR, which leads to RTO recovery quickly.

We also trace the congestion window depicted in Fig. 9. In this experiment, we set the receive buffer size larger than the maximum size of CWND in order to avoiding the effect of size of the remote advertised window on the send window.

Figure 9 plots the congestion window size with respect to the simulation time. From this figure, we can see that at about time 23 s when the FTP server of CN (i.e. the CN_FTP_Server)

starts to transmit the file, the size of CWND starts to increase exponentially. At time 26 s, the CWND rises up to 65,535 Bytes, which is the default value of Ssthresh, and TCP enters the congestion avoidance stage, where CWND increases linearly. Whenever the connection is broken during the handover, the standard TCP decreases the congestion windows size by one segment due to the timeout (see the curve of standard TCP/MIPv6 in Fig. 9). However, our approach does not invoke the slow start during the mobility over MIPv6 (see the curve of optimized TCP/MIPv6 in Fig. 9). Each time the handover finishes, the TCP CWND is reset according to the BDP [29] and the Ssthresh is equal to the CWND, which is usually higher than that of standard TCP. In this way, TCP prevents the invocation of any congestion control by estimating a suitable Ssthresh and CWND, and the congestion avoidance takes over.

5.3.2 Performance Measurements and Evaluation

Series of simulation experiments are also performed in examining the optimized TCP throughput performance for proposed TCP-CM. Figures 10 and 11 illustrate the effect of TCP-CM from the MN as TCP sender's and receiver's point of view respectively.

Figure 10 shows the average throughput when the CN_FTP_Server transfers a file to the MN_FTP_Client for performed experiments, from which the comparison of average throughput in a connection using our MIPv6-aware TCP and a connection using unmodified TCP can be observed. In this figure, the horizontal axis indicates the number of handover when the MN_FTP_Client moves around the wireless access networks, in which the endpoints are communicating with each other, and the vertical axis indicates the average throughput in terms of KBytes/Sec. The upper curve is the throughput of TCP with TCP-CM, and the bottom curve shows the throughput with standard TCP/MIPv6. As illustrated in this figure, when the number of handover increases, the TCP throughput will decline due to the frequent handovers. However, the overall throughput using TCP-CM is better than that of standard TCP/MIPv6. Moreover, the declining rate of TCP-CM is slower than that of standard TCP/MIPv6, which demonstrates that TCP-CM performs more outstanding when the handover occurs frequently. This is because that TCP-CM can detect the mobility events timely for any movement modes

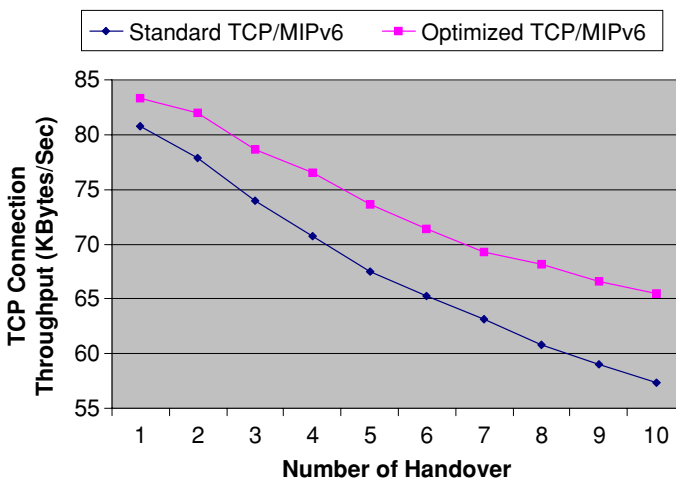


Fig. 10 Average throughput of TCP from CN sender's perspective

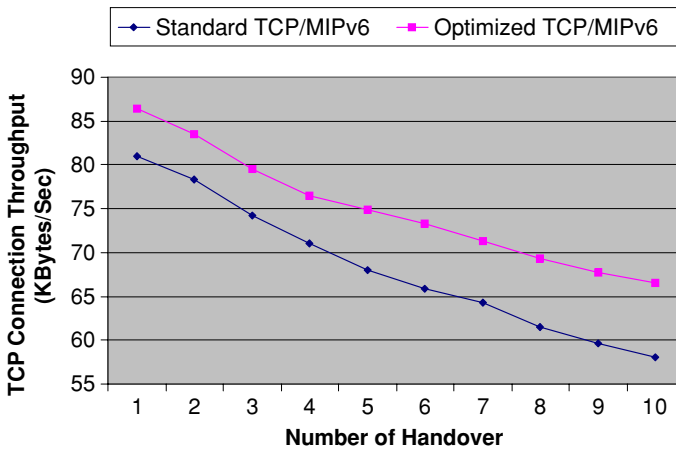


Fig. 11 Average throughput of TCP from MN sender's perspective

in frequent handovers, so that TCP-CM transmits data segments as fast as possible without waiting for an unnecessary timeout.

We also investigate the improvements of the throughput when the mobile is the TCP sender. Figure 11 depicts the average throughput versus the number of handover when the MN FTP_Server transfers a file to the CN FTP_Client for performed experiments, from which we can observe that the average throughput of TCP-CM will be still better than that of standard TCP/MIPv6. This is because that TCP-CM differentiates the mobility issue from the congestion issue and prevents TCP from invoking the unnecessary congestion control procedures.

6 Conclusions and Future Work

Like the network congestion, mobility is an important issue that TCP needs to care in mobile environments. In particular, the mobile Internet over MIPv6 suffers from the packet loss, deviation of transport delay and packet route changes, which are not due to the network congestion. We have shown how handover procedures of MIPv6 cause the packet loss and inflation of RTO and presented how the deviation of transport delay causes the RTT stale. We have also identified the factors that contribute to the performance degradation.

To adapt TCP to mobility better as well as improve the transport performance in networks over MIPv6, we have also proposed a new approach TCP-CM. The fast recovery and timing re-initialization mechanisms can eliminate the unnecessary waits for retransmission timeouts, and restore to a suitable network capability quickly; and the congestion avoidance mechanism prevents unnecessary invocation congestion control procedures after the MN moves to a new network, avoiding the degradation of throughput.

Our work makes clear that the need for TCP to deal with mobility control and congestion control respectively. Our results can be used to adapt TCP to the mobile Internet over MIPv6. Simulation results show that mobility control is significantly more robust than traditional TCP in the presence of mobile networks over MIPv6. Note TCP-CM is primarily designed as an extension to standard TCP (TCP Reno), which includes slow-start, congestion avoidance, fast retransmit and fast recovery. Clearly, although some other TCP variants such as

TCP Tahoe (which lacks of fast recovery from TCP Reno) [14] may not be better suitable in mobile environments than TCP Reno, it is still possible to apply the TCP-CM mechanisms described in this paper to improve the transport performance.

While our approach does deal with many of the transport performance problems associated with MIPv6, we still have further work needed to be done. For example, we do not consider the effect of some wireless links due to the presence of bursty wireless channel error, which however is in existence in the real wireless networks. In addition, since the MIPv6 is still not the perfect mobility solution, it is necessary to investigate the integration of our TCP-CM with the variants of MIPv6. In the future, we will study the effect of our approach in the real wireless networks with the presence of bursty wireless channel error, and we will integrate our TCP-CM with the MIPv6 extensions (i.e. HMIPv6 and FMIPv6) for further expected performance improvement of TCP.

Acknowledgements This work has been partially sponsored by the EU IST ENABLE research project. The use of OPNET Modeler in the research was facilitated through OPNET's university program. In addition, we would like to thank Fang-Chun Kuo, Ingo Juchem, Niklas Steinleitner, Jun Lei, and anonymous reviewers for their insightful comments to this paper.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

References

1. Allman, M., Paxson, V., & Stevens, W. (1999). TCP congestion control. In *RFC 2581*.
2. Antoine, S., Wei, M., & Aghvami, A. (2003). Impact of Mobile IPv6 handover on the performance of TCP: An experimental testbed. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(1), 31–33.
3. Bakre, A., & Badrinath, B. (1995). I-TCP: Indirect TCP for mobile hosts. In *Proceeding of 15th International Conference on Distributed Computing Systems (ICDCS'95)*.
4. Balakrishnan, H., Seshan, S., & Katz, R. (1995). Improving reliable transport and handoff performance in cellular wireless networks. *Wireless Networks*, 1(4), 469–481.
5. Border, J., Kojo, M., Griner, J., Montenegro, G., & Shelby, Z. (2001). Performance enhancing proxies intended to mitigate link-related degradations. In *RFC 3135*.
6. Brown, K., & Singh, S. (1997). M-TCP: TCP for mobile cellular networks. *ACM SIGCOMM Computer Communication Review*, 27(5), 19–43.
7. Caceres, R., & Iftode, L. (1995). Improving the performance of reliable transport protocols in mobile computing environments. *IEEE Journal on Selected Areas in Communications*, 13(5), 850–857.
8. Elaarag, H. (2002). Improving TCP performance over mobile networks. *ACM Computing Surveys*, 34(3), 357–374.
9. Goff, T., Moronski, J., Phatak, D., & Gupta, V. (2000). Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments. In *Proceedings of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'00)*.
10. Gurtov, A., & Korhonen, J. (2004). Effect of vertical handovers on performance of TCP-friendly rate control. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8(3), 73–87.
11. Ho, C., Chan, Y., & Chen, Y. (2005). An efficient mechanism of TCP-Vegas on mobile IP networks. In *Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*.
12. Hsieh, R., Seneviratne, A., Soliman, H., & El-Malki, K. (2002). Performance analysis on hierarchical Mobile IPv6 with fast-handoff over end-to-end TCP. In *Proceedings of Global Telecommunications Conference (GLOBECOM'02)*.
13. Izumikawa, H., Yamaguchi, I., & Katto, J. (2004). An efficient TCP with explicit handover notification for mobile networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC'04)*.

14. Jacobson, V. (1988). Congestion avoidance and control. In *ACM SIGCOMM'88* (pp. 314–329). Stanford, CA.
15. Johnson, D., Perkins, C., & Arkko, J. (2004). Mobility support in IPv6. In *RFC 3775*.
16. Kalman, M., & Girod, B. (2004). Modeling the delays of successively-transmitted Internet packets. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'04)*.
17. Karn, P., & Partridge, C. (1991). Improving round-trip time estimates in reliable transport protocols. *ACM Transactions on Computer Systems*, 9(4), 364–373.
18. Koodli, R. (2005). Fast handovers for Mobile IPv6. In *RFC 4068*.
19. Le, D., Fu, X., & Hogrefe, D. (2006). A review of mobility support paradigms for the Internet. *IEEE Communications Surveys and Tutorials*, 8(1), 2–15.
20. Matsushita, Y., Matsuda, T., & Yamamoto, M. (2005). Network supported bandwidth control for TCP in hierarchical mobile Internet. *IEICE Transactions on Communications*, E88-B(1), 266–273.
21. McNair, J., Tugcu, T., Wang, W., & Xie, J. (2005). A survey of cross-layer performance enhancements for mobile IP networks. *Computer Networks*, 49(2), 119–146.
22. Narten, T., Nordmark, E., & Simpson, W. (1998). Neighbor discovery for IP version 6 (IPv6). In *RFC 2461*.
23. OPNET Technologies, I. (2007). *OPNET Modeler*. OPNET Technologies.
24. Paxson, V., & Allman, M. (2000). Computing TCP's retransmission timer. In *RFC 2988*.
25. Postel, J. (1981). Transmission control protocol. In *RFC 793*.
26. Soliman, H. (2004). *Mobile IPv6: Mobility in a Wireless Internet*. Addison-Wesley.
27. Soliman, H., Castelluccia, C., El-Malki, K., & Bellier, L. (2005). Hierarchical Mobile IPv6 mobility management (HMIPv6). In *RFC 4140*.
28. Stangel, M., & Bhargavan, V. (1998). Improving TCP performance in mobile computing environments. In *Proceedings of International Conference on Communications (ICC'98)*.
29. Stevens, W. (1994). *TCP/IP Illustrated, Vol. 1: The Protocols*. Addison-Wesley.
30. Swami, Y., Le, K., & Eddy, W. (2006). Lightweight mobility detection and response (LMDR) algorithm for TCP. draft-swami-tcp-lmdr-07.txt, Internet Draft (work in progress).
31. Thomson, S., & Narten, T. (1998). IPv6 stateless address autoconfiguration. In *RFC 2462*.
32. Wehrle, K., Pählke, F., Ritter, H., Müller, D., & Bechler, M. (2004). *The Linux Networking Architecture: Design and Implementation of Network Protocols in the Linux Kernel*. Prentice Hall.

Author Biographies



Deguang Le (le@cs.uni-goettingen.de) received his Ph.D. Degree (with honors) from Xiamen University, China in 2006. He was a visiting scholar between 2004 and 2005 and a research assistant working on the EU FP6 IST ENABLE project in 2006 and 2007, both at the University of Göttingen. His research interests lie in IP mobility management protocols and their performance analysis.



Xiaoming Fu (fu@cs.uni-goettingen.de) is professor and head of Computer Networks Group at the University of Göttingen, Germany. He received his Ph.D. Degree in Computer Science from Tsinghua University, China in 2000. He was a research staff at Technical University Berlin, before moving to Göttingen as assistant professor in 2002. His research interests include network architectures, protocols, mobile communications and service overlays. He has served as TPC member/session chair/chair for several networking conferences such as INFOCOM, ICNP, ICDCS, and MobiArch. He is currently editorial board member of Elsevier Computer Communications Journal and a guest editor of IEEE Network Special Issue on Implications and Control of Middleboxes in the Internet.



Dieter Hogrefe received his Diploma Degree and Ph.D. from the University of Hannover, Germany. His research activities are directed towards Computer Networks and Protocol Engineering. In these fields he has published several books and numerous papers on Internet technology, analysis, simulation and testing of formally specified communication systems. After years of research positions in Siemens, he held professorships at the Universities of Dortmund, Berne and Luebeck. Since 2002 he is Professor for Telematics at the University of Göttingen. Since 1996 Prof. Hogrefe is chairman of the Technical Committee Methods for Testing and Specification at the European Telecommunication Standards Institute, ETSI.